Jon Postel

Network Working Group
Request For Comments: 1075

D. Waitzman
C. Partridge
BBN STC
S. Deering
Stanford University
November 1988

Distance Vector Multicast Routing Protocol

1. Status of this Memo

This RFC describes a distance-vector-style routing protocol for routing multicast datagrams through an internet. It is derived from the Routing Information Protocol (RIP) [1], and implements multicasting as described in RFC-1054. This is an experimental protocol, and its implementation is not recommended at this time. Distribution of this memo is unlimited.

2. Introduction

A draft standard for multicasting over IP networks now exists [2], but no routing protocols to support internetwork multicasting are available. This memo describes an experimental routing protocol, named DVMRP, that implements internetwork multicasting. DVMRP combines many of the features of RIP [1] with the Truncated Reverse Path Broadcasting (TRPB) algorithm described by Deering [3].

DVMRP is an "interior gateway protocol"; suitable for use within an autonomous system, but not between different autonomous systems. DVMRP is not currently developed for use in routing non-multicast datagrams, so a router that routes both multicast and unicast datagrams must run two separate routing processes. DVMRP is designed to be easily extensible and could be extended to route unicast datagrams.

DVMRP was developed to experiment with the algorithms in [3]. RIP was used as the starting point for the development because an implementation was available and distance vector algorithms are simple, as compared to link-state algorithms [4]. In addition, to allow experiments to traverse networks that do not support multicasting, a mechanism called "tunneling" was developed.

The multicast forwarding algorithm requires the building of trees based on routing information. This tree building needs more state information than RIP is designed to provide, so DVMRP is much more complicated in some places than RIP. A link-state algorithm, which already maintains much of the state needed, might prove a better basis for Internet multicasting routing and forwarding.

DVMRP differs from RIP in one very important way. RIP thinks in terms of routing and forwarding datagrams to a particular destination. The purpose of DVMRP is to keep track of the return paths to the source of multicast datagrams. To make explanation of DVMRP more consistent with RIP, the word "destination" is used instead of the more proper "source", but the reader must remember that datagrams are not forwarded to these destinations, but originate from them.

This memo is organized into the following sections:

- A description of DVMRP is presented.
- Tunnels are explained.
- The routing algorithm is shown.
- The forwarding algorithm is shown.
- The various time values are listed.
- Configuration information is specified.

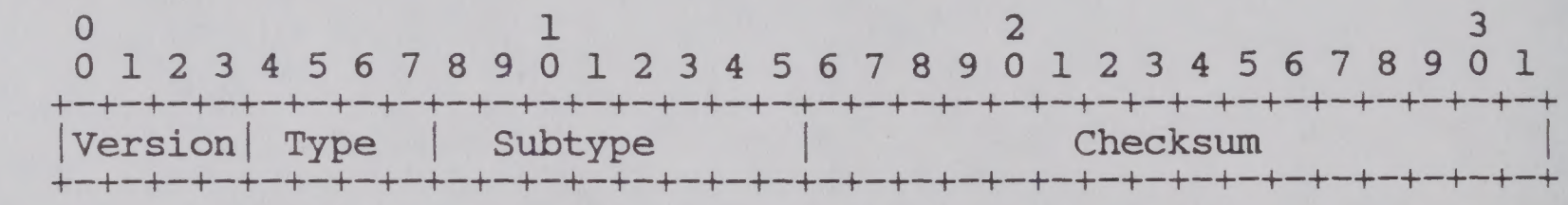
This memo does not analyze distance-vector routing, nor fully explain the distance-vector algorithm; see [1] for more information on these topics. The process or processes that perform the routing and forwarding functions are called "routers" in this memo.

3. Protocol Description

DVMRP uses the Internet Group Management Protocol (IGMP) to exchange routing datagrams [2].

DVMRP datagrams are composed of two portions: a small, fixed length IGMP header, and a stream of tagged data.

The fixed length IGMP header of DVMRP messages is:



The version is 1.

The type for DVMRP is 3.

The subtype is one of:

1 = Response; the message provides routes to some destination(s).
2 = Request; the message requests routes to some destination(s).
3 = Non-membership report; the message provides non-membership report(s).

4 = Non-membership cancellation; the message cancels previous non-membership report(s).

The checksum is the 16-bit one's complement of the one's complement sum of the entire message, excluding the IP header. For computing the checksum, the checksum field is zeroed.

The rest of the DVMRP message is a stream of tagged data. The reason for using a stream of tagged data is to provide easy extensibility (new commands can be developed by adding new tags) and to reduce the amount of redundant data in a message. The elements in the stream, called commands, are multiples of 16 bits, for convenient alignment. The commands are organized as an eight bit command numeric code, with at least an eight bit data portion. Sixteen-bit alignment of all commands is required.

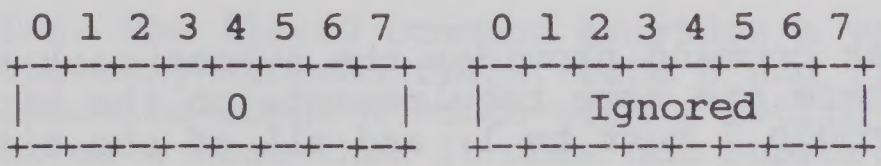
A message that has an error in it will be discarded at the point in processing where the error is detected. Any state changed due to the message contents before the error will not be restored to its previous values.

Certain commands have default values defined in their specification. As the defaults may be changed as the protocol is developed further, a cautious implementation will not send out messages that depend on defaults.

The length of DVMRP messages is limited to 512 bytes, excluding the IP header.

3.1 NULL Command

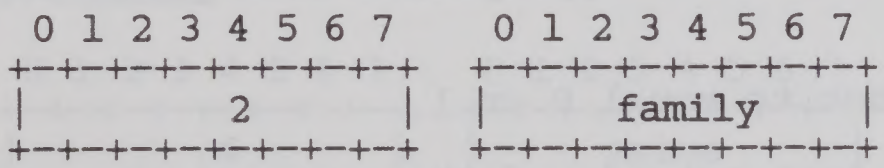
Format:



Description: The NULL command can be used to provide additional alignment or padding to 32 bits.

3.2 Address Family Indicator (AFI) Command

Format:



Values for family:

2 = IP address family, in which addresses are 32 bits long.

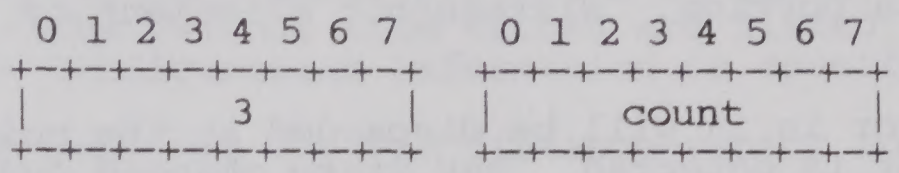
Default: Family = 2.

Description: The AFI command provides the address family for subsequent addresses in the stream (until a different AFI command is given).

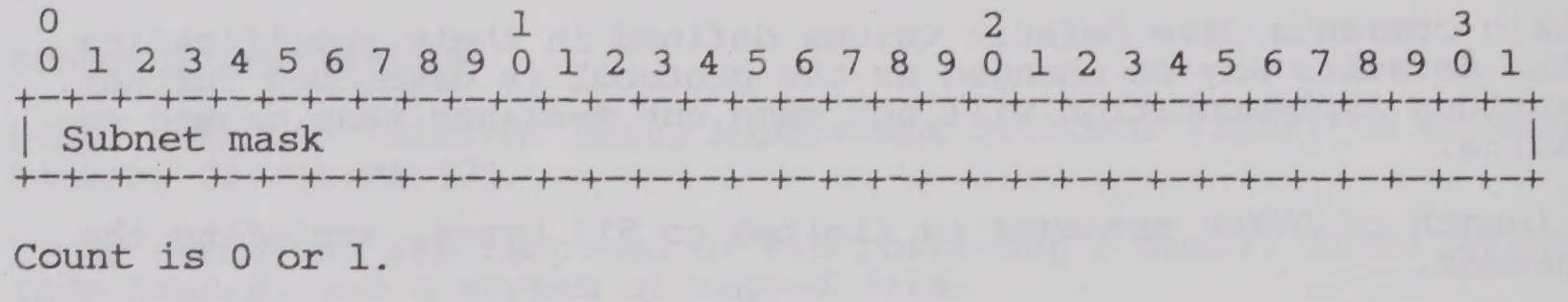
It is an error if the receiver does not support the address family.

3.3 Subnetmask Command

Format:



Additional argument, with AFI = IP:



Count is 0 or 1.

Default: Assume that following routes are to networks, and use a mask of the network mask of each route's destination.

Description: The Subnetmask command provides the subnet mask to use for subsequent routes. There are some requirements on the bits in the subnetmask: bits 0 through 7 must be 1, and all of the bits must not be 1.

If the count is 0, then no subnet mask applies, assume that the following routes are to networks, and use a mask of the network mask of each route's destination. If count is 1, then a subnet mask should be in the data stream, of an appropriate size given the address family.

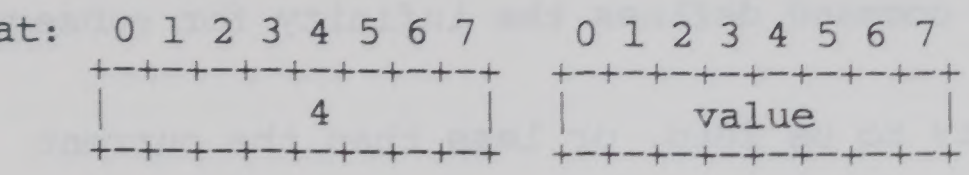
It is an error for count not to equal 0 or 1.

Subnetmasks should not be sent outside of the appropriate network.

See [6] for more information regarding IP subnetting.

3.4 Metric Command

Format:



Value is the metric, as an unsigned value ranging from 1 to 255.

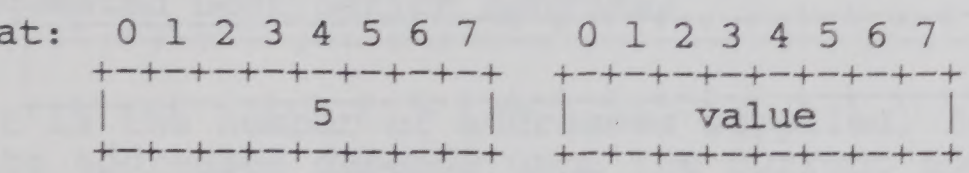
Default: None.

Description: The metric command provides the metric to subsequent destinations. The metric is relative to the router that sent this DVMRP routing update.

It is an error for metric to equal 0.

3.5 Flags0 Command

Format:



Meaning of bits in value:

Bit 7: Destination is unreachable.
Bit 6: Split Horizon concealed route.

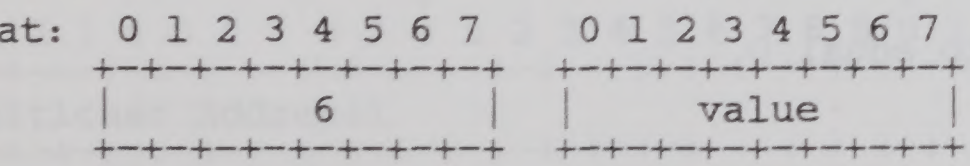
Default: All bits zero.

Description: The flags0 command provides a way to set a number of flags. The only defined flags, bits 6 and 7, can be used to provide more information about a route with a metric of infinity. A router that receives a flag that it does not support should ignore the flag. The command is called flags0 to permit the definition of additional flag commands in the future (flags1, etc.).

This is an experimental command, and may be changed in the future.

3.6 Infinity Command

Format:



Value is the infinity, as an unsigned value ranging from 1 to 255.

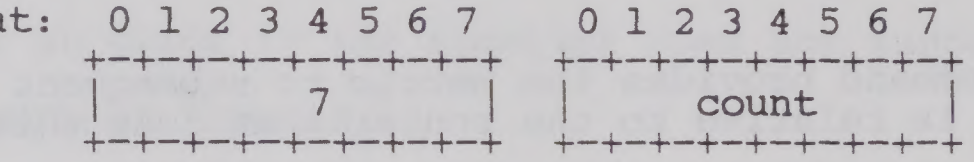
Default: Value = 16.

Description: The infinity command defines the infinity for subsequent metrics in the stream.

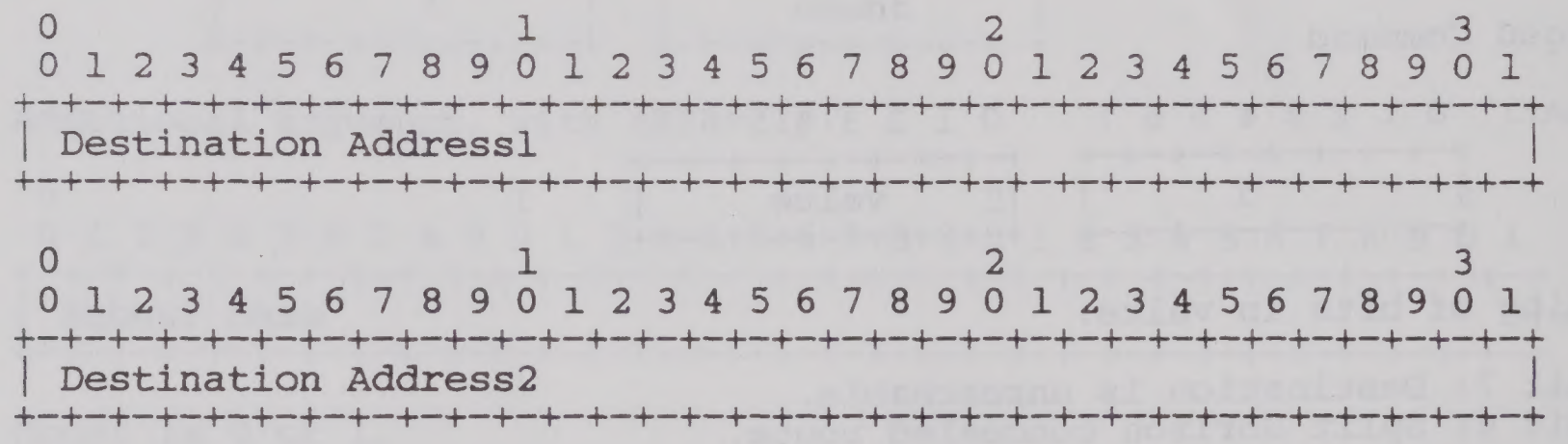
It is an error for infinity to be zero, or less than the current metric.

3.7 Destination Address (DA) Command

Format:



Array of 'count' additional arguments, with AFI = IP:



Count is the number of addresses supplied, from 1 to 255. The length of the addresses depends upon the current address family. The number of addresses supplied is subject to the message length limitation of 512 bytes.

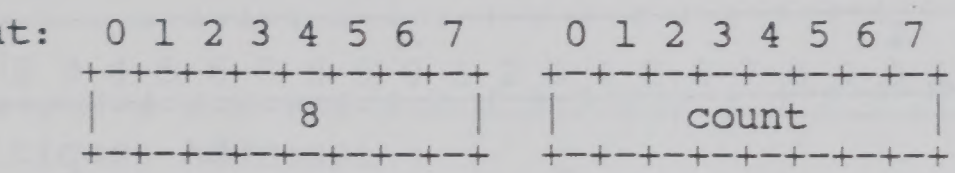
Default: None.

Description: The DA command provides a list of destinations. While this format can express routes to hosts, the routing algorithm only supports network and subnetwork routing. The current metric, infinity, flags0 and subnetmask, when combined with a single destination address, define a route. The current metric must be less than or equal to the current infinity.

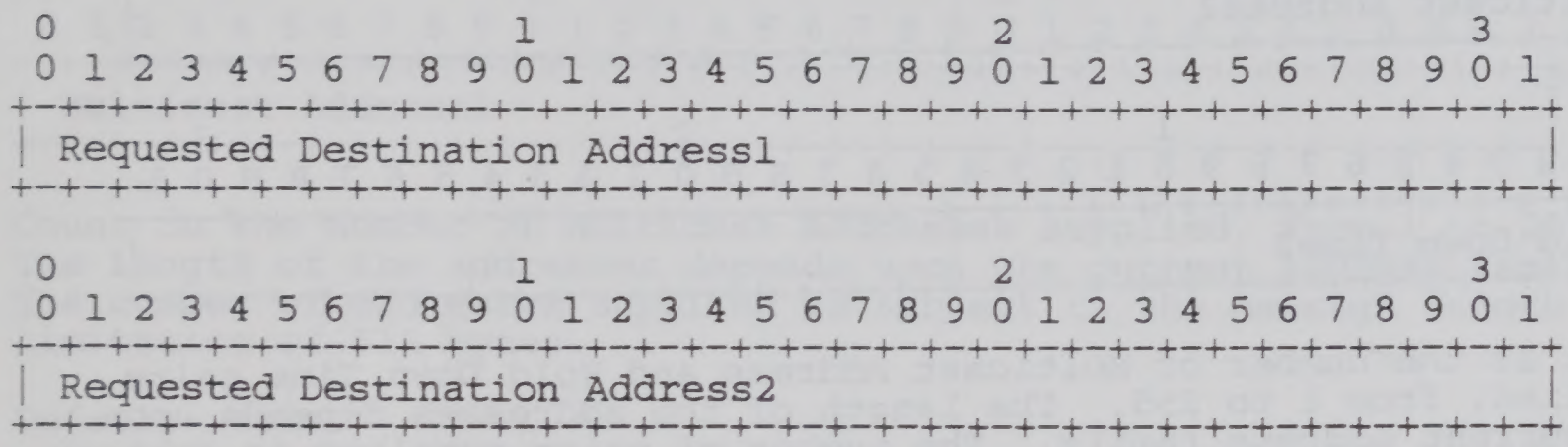
It is an error for count to equal 0.

3.8 Requested Destination Address (RDA) Command

Format:



Array of 'count' additional arguments, with AFI = IP:



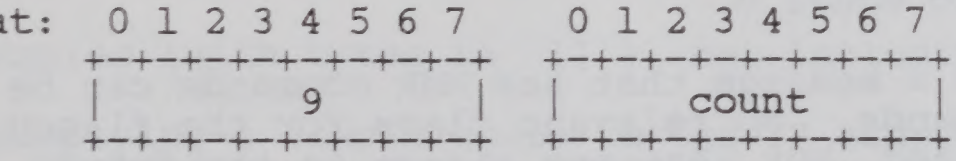
Count is the number of addresses supplied, from 0 to 255. The length of the addresses depends upon the current address family. The number of addresses supplied is subject to the message length limitation of 512 bytes.

Default: None.

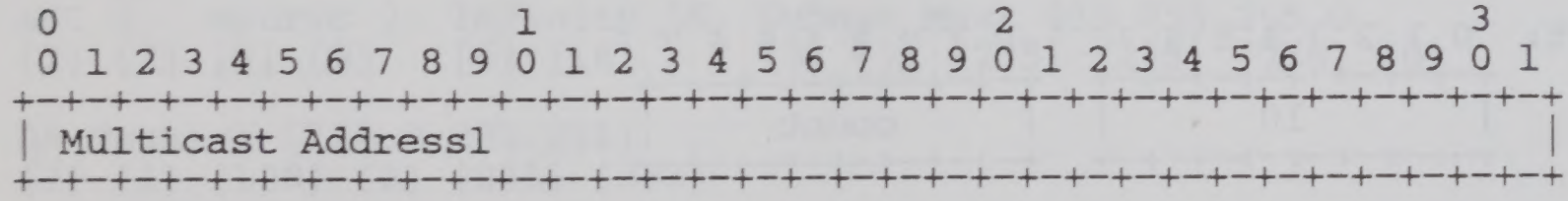
Description: The RDA command provides a list of destinations for whom routes are requested. A routing request for all routes is encoded by using a count = 0.

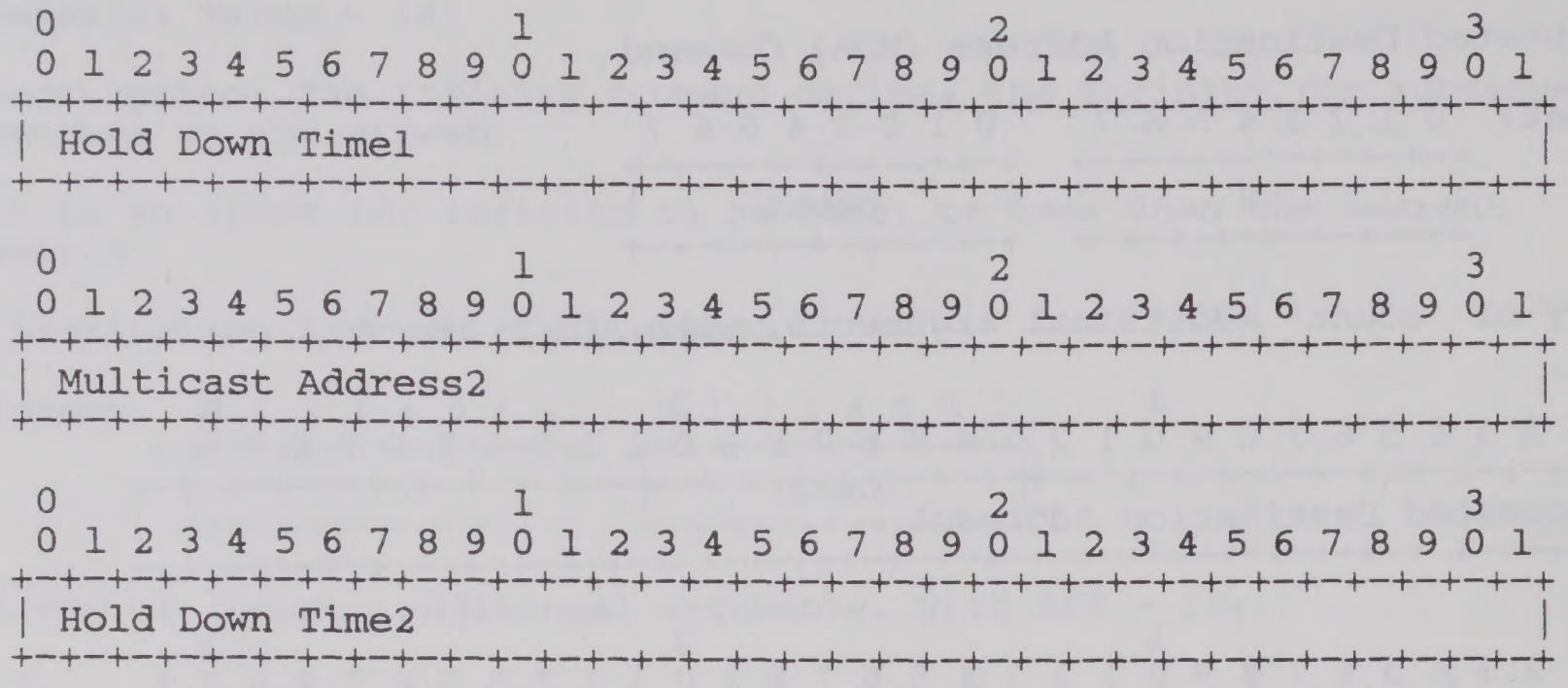
3.9 Non Membership Report (NMR) Command

Format:



Array of 'count' additional arguments, with AFI = IP:





Count is the number of Multicast Address and Hold Down Time pairs supplied, from 1 to 255. The length of the addresses depends upon the current address family. The number of pairs supplied is subject to the message length limitation of 512 bytes.

Default: None.

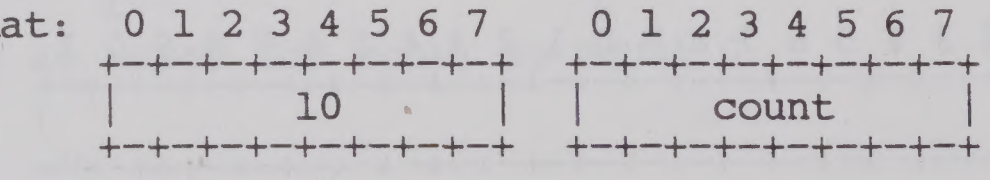
Description: The NMR command is experimental, and has not been tested in an implementation. Each multicast address and hold down time pair is called a non-membership report. The non-membership report tells the receiving router that the sending router has no descendent group members in the given group. Based on this information the receiving router can stop forwarding datagrams to the sending router for the particular multicast address(es) listed. The hold down time indicates, in seconds, how long the NMR is valid.

It is an error for count to equal 0.

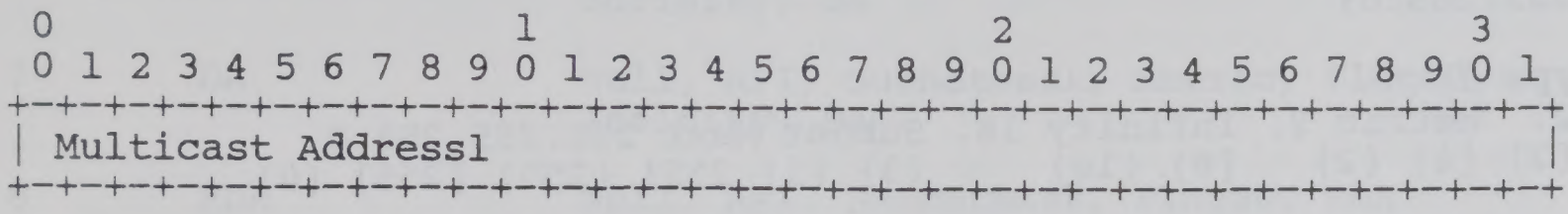
The only other commands in a message that has NMR commands can be the AFI, flags0, and NULL commands. No relevant flags for the flags0 command are currently defined, but that may change in the future.

3.10 Non Membership Report Cancel (NMR Cancel) Command

Format:



Array of 'count' additional arguments, with AFI = IP:



Count is the number of Multicast Addresses supplied, from 1 to 255. The length of the addresses depends upon the current address family. The number of addresses supplied is subject to the message length limitation of 512 bytes.

Default: None.

Description: The NMR Cancel command is experimental, and has not been tested in an implementation. For each multicast address listed, any previous corresponding non-membership reports are canceled. When there is no corresponding non-membership report for a given multicast address, the Cancel command should be ignored for that multicast address.

It is an error for count to equal 0.

The only other commands in a message that has NMR Cancel commands can be the AFI, flags0, and NULL commands. No relevant flags for the flags0 command are currently defined, but that may change in the future.

3.12 Examples (with bytes in '{}'), not including the message header:

3.12.1 Supplying a single route to the IP address 128.2.251.231 with a metric of 2, an infinity of 16, a subnetmask of 255.255.255.0:

Subtype 1,
AFI 2, Metric 2, Infinity 16, Subnet Mask 255.255.255.0
{2} {2} {4} {2} {6} {16} {3} {1} {255} {255} {255} {0}

DA Count=1 [128.2.251.231]
{7} {1} {128} {2} {251} {231}

3.12.2 Supplying a route to the IP addresses 128.2.251.231 and 128.2.236.2 with a metric of 2, an infinity of 16, a subnetmask of 255.255.255.0:

```
Subtype 1,
AFI 2,  Metric 2, Infinity 16, Subnet Mask 255.255.255.0
{2} {2} {4} {2}   {6} {16}     {3} {1} 255} {255} {255} {0}

DA Count=2 [128.2.251.231] [128.2.236.2]
{7} {1} {128} {2} {251} {231} {128} {2} {236} {2}
```

3.12.3 Request for all routes to IP destinations.

```
Subtype 2, AFI 2,  RDA Count = 0
           {2} {2} {8} {0}
```

3.12.4 Non Membership Report for groups 224.2.3.1 and 224.5.4.6 with a hold down time of 20 seconds, and group 224.7.8.5 with a hold down time of 40 seconds.

```
Subtype 3,
AFI 2,  NMR Count = 3 [224.2.3.1, 20]
{2} {2} {10} {3} {224} {2} {3} {1} {0} {0} {0} {20}

[224.5.4.6, 20] [224.7.8.5, 40]
{224} {5} {4} {6} {0} {0} {0} {20} {224} {7} {8} {5} {0} {0} {0} {40}
```

3.13 Summary of Commands

Value	Name	Other commands allowed in same message
0	Null	Null, AFI, Subnetmask, Metric, Flags0, Infinity, DA, RDA, NMR, NMR-cancel
2	AFI	Null, AFI, Subnetmask, Metric, Flags0, Infinity, DA, RDA, NMR, NMR-cancel
3	Subnetmask	Null, AFI, Subnetmask, Metric, Flags0, Infinity, DA, RDA
4	Metric	Null, AFI, Subnetmask, Metric, Flags0, Infinity, DA
5	Flags0	Null, AFI, Subnetmask, Metric, Flags0, Infinity, DA

6	Infinity	Null, AFI, Subnetmask, Metric, Flags0, Infinity, DA
7	DA	Null, AFI, Subnetmask, Metric, Flags0, Infinity, DA
8	RDA	Null, AFI, Subnetmask, Flags0, RDA
9	NMR	Null, AFI, Flags0, NMR
10	NMR-cancel	Null, AFI, Flags0, NMR-cancel

4. Tunnels

A tunnel is a method for sending datagrams between routers separated by gateways that do not support multicasting routing. It acts as a virtual network between two routers. For instance, a router running at Stanford, and a router running at BBN might be connected with a tunnel to allow multicast datagrams to traverse the Internet. We consider tunnels to be a transitional hack.

Tunneling is done with a weakly encapsulated normal multicasted datagram. The weak encapsulation uses a special two element IP loose source route [5]. (This form of encapsulation is preferable to "strong" encapsulation, i.e., prepending an entire new IP header, because it does not require the tunnel end-points to know each other's maximum reassembly buffer size. It also has the benefit of correct behavior of the originator's time-to-live value and any other IP options present.)

A tunnel has a local end-point, remote end-point, metric, and threshold associated with it. The routers at each end of the tunnel need only agree upon the local and remote end-points. See section 8 for information on how tunnels are configured. Because the number of intermediate gateways between the end-points of a tunnel is unknown, additional research is needed to determine appropriate metrics and thresholds.

To send a datagram on a tunnel, the following occurs:

- A null IP option is inserted into the datagram. This provides preferred alignment for the loose source route IP option.

- A two element loose source route IP option is inserted into the datagram.

- The source route pointer is set to point to the second element

in the source route.

- The first element in the source route is replaced with the address of the originating host (the original IP source address).

- The second element in the source route is replaced with the multicast destination address provided by the originating host (the original IP destination address).

- The IP source address is replaced with the address of the router's appropriate outgoing physical interface (the local tunnel end-point).

- The IP destination address is replaced with an address of the remote router (the remote tunnel end-point).

- The datagram is transmitted to the remote router using non-multicast routing algorithms.

Intermediate, non-multicast gateways will route the tunneled datagram to the remote tunnel end-point. Because the datagram's IP source address has been replaced with the address of the local tunnel end-point, ICMP error messages will go to the originating multicast router. This behavior is desired, because a host that sends a multicast datagram, which a multicast router decides to tunnel, should not be aware of the use of the tunnel. If the datagram's IP source address were not changed when encapsulating the datagram, any ICMP errors would be sent to the originating host.

When the remote tunnel end-point receives the tunneled datagram, the following occurs:

- The IP source address is replaced with the first element in the loose source route.

- The IP destination address is replaced with the second element in the loose source route.

- The null option and the loose source route option are removed from the datagram. This is needed because a host should not be able to tell that it has received a datagram that was sent through a tunnel.

Because no specific network is associated with a tunnel, there are no local group memberships to be tracked for a tunnel. The only neighbor on a tunnel can be the remote end-point. Routing messages should be exchanged through tunnels, but a route is not created for a

tunnel. The routing messages should be sent as unicast datagrams directly to the remote tunnel end-point; they should not use an IP loose source route.

Justification for using the loose source route and record option for tunneling:

We considered defining our own IP option to handle tunneling, but we are worried that intermediate gateways do not transparently pass IP options that are unknown to them. Datagrams using a new option would not traverse the Internet. It would be better for us if we could create a new IP option, but it won't work presently. Recall that this is a transition design to allow us to experiment in the current environment.

The tunneled packet containing the LSRR option has the following features:

Field	Value
src address	= src gateway address
dst address	= dst gateway address
LSRR pointer	= points to LSRR address 2
LSRR address 1	= src host
LSRR address 2	= multicast destination

Two questions raised about using the LSRR option for tunnels are "Can intermediate gateways ignore the option?", and "Can the destination gateway properly detect that the LSRR is used for a tunnel?".

When an intermediate gateway receives a datagram, it examines the destination address. For a tunneled datagram, the destination address will not match an address of the receiving gateway. Therefore, the LSRR option will not be examined, and the intermediate gateway will forward the datagram on to its next hop for the destination address.

When the destination gateway receives a datagram, it notes that the datagram destination address matches one of its own address. It will then look at the next LSRR option address, since the source route is not exhausted. That address is a multicast address. Since hosts are forbidden from putting multicast addresses into source routes, the gateway can infer that the LSRR is for tunneling. The weakness here is that perhaps there is some other meaning for the multicast address in the LSRR. No other meaning is currently defined.

If a tunneled datagram is by error addressed to a destination gateway that does not support multicasting, then the destination gateway will try to find a route to the multicast address. This will fail, and an ICMP destination unreachable error message will be sent to the tunneled datagram's source. Since the source address in the tunneled datagram has been adjusted to be the address of the source multicast gateway, the ICMP errors will not go to the originating host, which has no knowledge of tunnels.

5. Routing Algorithm

This section provides a terse description of the distance-vector routing algorithm. See [1] for more information.

While DVMRP can express routes to individual hosts, the forwarding and routing algorithms only support network and subnetwork routing.

In the discussion below, the term "virtual interface" is used to refer to a physical interface or a tunnel local end-point. A physical interface is a network interface, for instance, an Ethernet card. A route to a destination will be through a virtual interface. The term "virtual network" is used to refer to a physical network or a tunnel, with the qualification that routes only reference physical networks.

The TRPB algorithm forwards multicast datagrams by computing the shortest (reverse) path tree from the source (physical) network to all possible recipients of the datagram. Each multicast router must determine its place in the tree, relative to the particular source, and then determine which of its virtual interfaces are in the shortest path tree. The datagram is forwarded out these virtual interfaces. The process of excluding virtual interfaces not in the shortest path tree is called "pruning."

Consider a virtual network. Using Deering's terminology [3], a router is called the "parent" of the virtual network if that router is responsible for forwarding datagrams onto that virtual network through its connecting virtual interface. The virtual network can also be considered a "child" virtual network of the router. Using the child information, routers can do Reverse Path Broadcasting (RPB).

Unnecessary datagrams may still be sent onto some networks, because there might not be any recipients for those datagrams on the networks. There are two kinds of recipients: hosts that are members of a particular multicast group, and multicast routers. If no multicast routers on a virtual network consider that virtual network uptree to a given source, then that virtual network is a "leaf"

network. If a network is a leaf for a given source, and there are no members of a particular group on the network, then there are no recipients for datagrams from the source to the group on that network. That network's parent router can forgo sending those datagrams on that network, or "truncate" the shortest path tree. The algorithm that tracks and uses this information is the Truncated Reverse Path Broadcasting (TRPB) algorithm.

Determining which virtual networks are leaves is not simple. If any neighboring router considers a given virtual network in the path to a given destination, then the virtual network is not a leaf. Otherwise, it is a leaf. This is a voting function. If a route, with a metric poisoned by split horizon processing, is sent by some router, then that router uses that virtual network as the uptree path for that route (i.e. that router votes that the virtual network is not a leaf relative to the route's destination). Since the number of routers on a virtual network is dynamic, and since all received routing updates are not kept by routers, a heuristic is needed to determine when a network is a leaf. DVMRP samples the routing updates on a virtual interface while a hold down timer is running, which is for a time period of LEAF_TIMEOUT seconds. There is one hold down timer per virtual interface. If a route is received with a metric poisoned by split horizon processing while the hold down timer is running, or at any other time, then the appropriate virtual interface for that route is "spoiled"-- it is not a leaf. For every route, any virtual interface that was not spoiled by the time the hold down timer expires is considered a leaf.

For a description of an even better forwarding algorithm, the Reverse Path Multicasting algorithm, see [3].

A route entry should have the following in it:

- Destination address (a source of multicast datagrams) *
- Subnet mask of the destination address *
- Next-hop router to the destination address
- Virtual interface to the next-hop router *
- List of child virtual interfaces *
- List of leaf virtual interfaces *
- A dominant router address for each virtual interface
- A subordinate router address for each virtual interface
- Timer
- Set of flags that indicate the state of the entry
- Metric
- Infinity

The lines that are marked with '*' indicate fields that are directly used by the forwarding algorithm.

The lists of child and leaf interfaces can be implemented as bitmaps.

5.1 Sending Routing Messages

DVMRP routing messages can be used for three basic purposes: to periodically supply all routing information, to gratuitously supply routing information for recently changed routes, or supply some or all routes in response to a request.

Routing messages sent to physical interfaces should have an IP TTL of 1.

A number of timeouts and rates are used by the routing and forwarding algorithms. See section 6 for their values.

Rules for when to send routing messages:

- Every FULL_UPDATE_RATE seconds a router should send out DVMRP messages with all of its routing information to all of its virtual interfaces. To prevent routers from synchronizing when they send updates, a real-time timer must be used.

- Whenever a route is changed, a routing update should be sent for that route. Some delay must occur between triggered updates to avoid flooding the network with triggered updates; intervals of TRIGGERED_UPDATE_RATE seconds is suggested.

- A request for all routes should be sent on all virtual interfaces when an DVMRP router is restarted.

- If possible, when a DVMRP router is about to terminate execution, it should send out DVMRP messages with metrics equal to infinity for all of its routes, on all virtual interfaces.

When sending to routers connected via networks that support multicasting, the messages should be multicast to address 224.0.0.4. Therefore, routers must listen to multicast address 224.0.0.4 on every physical interface that supports multicasting. If multicasting isn't supported, broadcasting can be used. As already mentioned, routing updates to tunnels should be sent as unicast datagrams to the remote end-point of the tunnel.

When sending routing messages, except in response to a specific route request (via RDA command with a non-zero count), poisoned split horizon processing must be done. This means that given a route that uses network X, routing updates sent to network X must include that route with the metric equal to the infinity and should include the

appropriate flag set in a FLAGS0 command.

Poisoned split horizon is one way to reduce the likelihood of routing loops. Another method, not available in RIP, is to choose a better infinity in a route. For routes propagated in a small, but well connected, network an infinity smaller than 16 might be better. The smaller the infinity, the less time a counting-to-infinity event will take. In traversing a wide internet, an infinity of 16 might be too small. At the cost of a longer counting-to-infinity event, the infinity can be increased.

One concept in Internet Multicasting is to use "thresholds" to restrict which multicast datagrams exit a network. Multicast routers on the edge of a subnetted network or autonomous system may require a datagram to have large TTL to exit a network. This mechanism keeps most multicast datagrams within the network, reducing external traffic. An application that wants to multicast outside of its network would need to give its multicast datagrams at least a TTL of the sum of the threshold and the distance to the edge of the network (assuming TTL is used as a hop count within the network). A configuration option should allow specifying the threshold for both physical interfaces and tunnels.

When a router is started, it must send out a request for all routes on each of its virtual interfaces. The request is a message that has an RDA command with a count equal to 0 in it.

5.2 Receiving Routing Messages

A router must know the virtual interface that a routing message arrived on. Because the routing message may be addressed to the all-multicast-routers IP address, and because of tunnels, the incoming interface can not be identified merely by examining the message's IP destination address

For each route expressed in a routing message, the following must occur:

```
IF a metric was given for the route:
THEN    add in the metric of the virtual interface that the message
        arrived on.

Lookup the route's destination address in the routing tables.

IF the route doesn't exist in the tables:
THEN    try to find a route to the same network in the routing
        tables.
        IF that route exists in the tables:
```

```
            THEN    IF this route came from the same router as the router
                    that the found route came from:
                    THEN    CONTINUE with next route.
            IF route doesn't have a metric of infinity:
                    THEN    add the route to the routing tables.
            CONTINUE with next route.

    IF this route came from the same router as the router that the found
       route came from:
    THEN    clear the route timer.
            IF a metric was received, and it is different than the found
            route's metric:
            THEN    change the found route to use the new metric and
                    infinity.
                    IF the metric is equal to the infinity:
                    THEN    set the route timer to the
                            EXPIRATION_TIMEOUT.
                    CONTINUE with next route.
            IF the received infinity does not equal the found route's
            infinity:
            THEN    change the found route's infinity to be the received
                    infinity.
                    change the found route's metric to be the minimum of
                    the received infinity and the found route's metric.
    ELSE    IF a metric was received, and (it is less than the found
            route's metric or (the route timer is at least halfway to the
            EXPIRATION_TIMEOUT and the found route's metric equals the
            received metric, and the metric is less than the received
            infinity)):
            THEN    change the routing tables to use the received route.
                    clear the route timer.
    CONTINUE with next route.
```

5.3 Neighbors

A list should be kept of the neighboring multicast routers on every attached network. The information can be derived by the DVMRP routing messages that are received. A neighbor that has not been heard from in NEIGHBOR_TIMEOUT seconds should be considered to be down.

5.4 Local Group Memberships

As required by [2], a multicast router must keep track of group memberships on the multicast-capable networks attached to it. Every QUERY_RATE seconds an IGMP membership request should be sent to the All Hosts multicast address (224.0.0.1) on each network by a designated router on that network. The IGMP membership request will

cause hosts to respond with IGMP membership reports after a small delay. Hosts will send the report for a group to the group's multicast address.

The membership requests should have an IP TTL of 1.

The routers on a network elect or "designate" a single router to do the queries. The designated router is the router with the lowest IP address on that network. Upon startup a router considers itself to be the designated router until it learns (presumably through routing messages) of a router with a lower address. To learn about the group members present on a network at startup, a router should multicast a number of membership requests, separated by a small delay. We suggest sending three requests separated by four seconds.

The multicast router must receive all datagrams sent to all multicast addresses. Upon receiving an IGMP membership report for a group from an interface, it must either record the existence of that group on the interface and record the time, or update the time if the group is already recorded. The recorded group memberships must be timed-out. If a group member report is not received for a recorded group after MEMBERSHIP_TIMEOUT seconds, the recorded group should be deleted.

6. Forwarding Algorithm

The section describes the multicast forwarding algorithm and the state that must be kept for the algorithm.

The forwarding algorithm is applied to determine how multicast datagrams arriving on a physical interface or a tunnel should be handled. If multicast datagrams were flooded, a datagram received on one virtual interface would be forwarded out of every other virtual interface. Because of redundant paths in the internet, datagrams would be duplicated. The child and leaf information, that the routing algorithm supplies, is used to prune branches in the tree to all possible destinations.

In route entries, there is a dominant router address for each virtual interface. This address is the address of some router that has a route with a lower metric (and whose metric does not equal infinity) to the destination, on that virtual interface. The dominant router address is not set for the next-hop virtual interface.

Also in route entries, there is a subordinate router address for each virtual interface. This address is the address of some router that considers this router to be the parent of the virtual network. Therefore, the subordinate router address is not set for a virtual interface to a leaf network.

The algorithm for manipulating the children and leaf lists in route entries is:

Upon router startup:
Create a route entry for each virtual interface, with:
- all other virtual interfaces in its child list,
- an empty leaf list,
- no dominant router addresses, and
- no subordinate router addresses.

Start a hold down timer for each virtual interface, with a value of LEAF_TIMEOUT.

Upon receiving a new route:
Create the route entry, with:
- all virtual interfaces, other than the one on which the new route was received, in its child list,
- empty leaf list,
- no dominant router addresses, and
- no subordinate router addresses.

Start the hold down timer for all virtual interfaces, other than the one on which the new route was received, with a value of LEAF_TIMEOUT.

Upon receiving a route on virtual interface V from neighbor N with a lower metric than the one in the routing table (or the same metric as the one in the routing table, if N's address is less than my address for V), for that route:
If V is in the child list, delete V from the child list.
If there is no dominant router for V and if V is not (now) the next-hop virtual interface, record N as the dominant router.

Upon receiving a route on virtual interface V from neighbor N with a larger metric than the one in the routing table (or the same metric as the one in the routing table, if N's address is greater than my address for V), for that route:
If N is the dominant router for V, delete N as the dominant router and add V to the child list.

Upon receiving a route from neighbor N on virtual interface V with a metric equal to infinity (the split horizon flag should also be set), for that route:
If V is in the leaf list, delete V from the leaf list.
If there is no subordinate router for V, record N as the subordinate router.

Upon receiving a route from neighbor N on virtual interface V with a metric other than infinity (and no split horizon flag), for that route:

If N is the subordinate router for V, delete N as the subordinate router and start the hold down timer for V.

Upon timer expiration for a virtual interface (V), for each route:
If there is no subordinate router for V, add V to the leaf list.

Upon failure of neighbor N on virtual interface V, for each route:
If N is the dominant router for V, delete N as the dominant router and add V to the child list.
If N is the subordinate router for V, delete N as the subordinate router and start the hold down timer for V.

The forwarding algorithm is:

```
IF the IP TTL is less than 2:
THEN    CONTINUE with next datagram.

find the route to the source of the IP datagram.

IF no route exists:
THEN    CONTINUE with next datagram.

IF the datagram was not received on the next-hop virtual interface
for the route:
THEN    CONTINUE with next datagram.

IF the datagram is tunneled:
THEN    replace the datagram's source address with the first address
        in the IP loose source route.
        replace the datagram's destination address with the second
        address in the IP loose source route.
        delete the loose source route and the null option from the
        datagram and adjust the IP header length fields to reflect
        the deletion.

If the datagram destination is group 224.0.0.0 or group 224.0.0.1:
THEN    CONTINUE with next datagram.

FOR each virtual interface V
DO      IF V is in the child list for the source of the datagram:
        THEN    IF V is not in the leaf list for the source
                OR there are members of the destination group on V:
                THEN    IF the IP TTL is greater then V's threshold:
                        THEN    subtract 1 from the IP TTL
                                forward the datagram out V
```

7. Time Values

This section contains a list of the various rates and timeouts, their meanings, and their values. All values are in seconds.

How dynamic the routing environment is effects the following rates. A lower rate will allow quicker adaptation to a change in the environment, at the cost of wasting network bandwidth.

FULL_UPDATE_RATE = 60
- How often routing messages containing complete routing tables are sent.

TRIGGERED_UPDATE_RATE = 5
- How often triggered routing messages may be sent out.

Raising the following rates and timeouts may increase the time that packets may be forwarded to a virtual interface unnecessarily.

QUERY_RATE = 120
- How often local group membership is queried.

MEMBERSHIP_TIMEOUT = 2 * QUERY_RATE + 20
- How long a local group membership is valid without confirmation.

LEAF_TIMEOUT = 2 * FULL_UPDATE_RATE + 5
- How long the hold down timer is for a virtual interface.

Increasing the following timeouts will increase the stability of the routing algorithm, at the cost of slower reactions to changes in the routing environment.

NEIGHBOR_TIMEOUT = 4 * FULL_UPDATE_RATE
- How long a neighbor is considered up without confirmation. This is important for timing out routes, and for setting the children and leaf flags.

EXPIRATION_TIMEOUT = 2 * FULL_UPDATE_RATE
- How long a route is considered valid without confirmation. When this timeout expires, packets will no longer be forwarded on the route, and routing updates will consider this route to have a metric of infinity.

GARBAGE_TIMEOUT = 4 * FULL_UPDATE_RATE
- How long a route exists without confirmation. When this timeout expires, routing updates will no longer contain any information on this route, and the route will be deleted.

8. Configuration options

A router should be configurabled with the following information:

- Tunnel descriptions: local end-point, remote end-point, metric, and threshold. If no threshold is provided, the metric should be used as the default threshold.

- For a physical interface: metric, infinity, threshold and subnetwork mask. If no threshold is provided, the metric should be used as the default threshold.

9. Conclusion

This memo has presented DVMRP, an extensible distance-vector-style routing protocol, and a TRPB routing algorithm. An implementation of the ideas presented in this document has been done, and is being tested.

The added features in DVMRP, as compared to RIP, give it flexibility at the cost of more complex processing. DVMRP still has the disadvantages of being a distance-vector algorithm. Because link-state algorithms maintain much of the state information that DVMRP has to maintain in excess of what RIP needs, a multicast link-state routing protocol should be developed.

The TRPB algorithm can cause unneeded datagrams to be sent. The Reverse Path Multicasting algorithm (RPM) [3] might be a better algorithm. The NMR and NMR-cancel DVMRP messages are designed to support RPM. Further research is needed on this topic.

10. Acknowledgements

We would like to thank Robb Foster, Alan Dahlbom, Ross Callon, and the IETF Host Working Group for their ideas.

11. Bibliography

[1] Hedrick, C., "Routing Information Protocol", RFC 1058, Rutgers University, June 1988.

[2] Deering, S., "Host Extensions for IP Multicasting", RFC 1054, Stanford University, May 1988.

[3] Deering, S., "Multicast Routing in Internetworks and Extended LANs", SIGCOMM Summer 1988 Proceedings, August 1988.

[4] Callon, R., "A Comparison of 'Link State' and 'Distance

Vector' Routing Algorithms", DEC, November 1987.

[5] Postel, J., "Internet Protocol", RFC 791, USC/Information Sciences Institute, September 1981.

[6] Mills, D., "Toward an Internet Standard Scheme for Subnetting", RFC 940, University of Delaware, April 1985.